インテル® System Studio 2018 for Windows* ホスト/Linux* ターゲット入門ガイド

このガイドは、2018 年 2 月 16 日時点の、インテル® デベロッパー・ゾーンに公開されている「Intel® System Studio 2018 Getting Started Guide for Windows OS Host and Linux OS Target」の日本語訳です。

インテル® System Studio 2018 - 概要

インテル® System Studio 2018 は、デバイスメーカー、システム・インテグレーター、組込みアプリケーション開発者に、インテル® アーキテクチャー・ベースのソリューションを作成するために使用できるソフトウェア開発ツールの包括的なセットを提供します。インテルの拡張機能を Eclipse* ベースの IDE に統合して使用し、開発サイクルのすべてのフェーズで一貫した作業を行うことができます。

インテル® System Studio 2018 には、目的に応じて 3 つのエディションが用意されています。

- Composer Edition: アプリケーションの開発
- Professional Edition: アプリケーションの開発、プロファイルおよび解析
- Ultimate Edition: アプリケーションの開発、プロファイルおよび解析、ハードウェア・システム/プラットフォームの起動

IoT (Internet of Things) 開発者の場合、または Linux* ターゲット向けにコンテナーでプロジェクトをビルドしている場合は、『インテル® System Studio 2018 Gold ユーザーガイド: コンテナーでビルドして Linux* で実行する C/C++ プロジェクト』(英語) の説明を参照してください。

この入門ガイドは以下の 3 つのセクションから構成されています。

- アプリケーション開発
- プロファイルおよび解析
- ハードウェア・システム/プラットフォームの起動

インテル® System Studio 2018 for Windows* ホスト/Linux* ターゲットを使用したアプリケーション開発

ビルド・コンポーネントの概要

- インテル® C++ コンパイラー: Eclipse* IDE で最適化されたアプリケーションを開発
- インテル® マス・カーネル・ライブラリー (インテル® MKL): アプリケーションのパフォーマンスを向上
- インテル® スレッディング・ビルディング・ブロック (インテル® TBB): マルチコア・アーキテクチャー向け組込みソリューションを最適化
- インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP): 信号/画像処理、データ圧縮、ストリング操作などをスピードアップ
- インテル® データ・アナリティクス・アクセラレーション・ライブラリー (インテル® DAAL): マシンラーニングおよびビッグデータ解析のパフォーマンスを高速化
- GNU* プロジェクト・デバッガー (GDB) アプリケーション・デバッガー - Linux* ターゲットのみ: アプリケーションのクラッシュの原因を正確に特定

Eclipse* での Windows* ホスト/Linux* ターゲット・アプリケーションの開発

インテル® System Studio には、インテル® C++ コンパイラーを使用して Windows* ホスト/Linux* ターゲット・アプリケーションを開発する、2 つの Eclipse* ベースのワークフローが用意されています。

- **Linux* および Android* ターゲット向けクロスコンパイル・プロジェクト:** プロジェクト環境をセットアップして独自のコンパイルおよびビルドツールを設定します。このガイドで説明しているワークフローです。
- **コンテナーでビルドして Linux* で実行するプロジェクト:** Docker* コンテナー形式で設定されたコンパイルおよびビルドツールを提供します。これらのワークフローは、高速な編集、デバッグ、配備プロセスを使用して Linux* ベースの IoT アプリケーションを作成する場合に非常に役立ちます。Docker* コンテナーの使用に関する情報は、『インテル® System Studio 2018 ユーザーガイド: コンテナーでビルドして Linux* で実行する C/C++ プロジェクト』(英語) を参照してください。

作業を進める前に、次の準備を完了してください。準備が完了したら、「Linux* ターゲット向けクロスコンパイル・プロジェクトの作成」セクションに進んでください。

準備

root 権限を使用したターゲット接続の作成

MRAA/UPM ライブラリーを使用するアプリケーションは、ハードウェア (GPIO、I2C および SPI バス) を制御できるように root 権限で実行する必要があります。このセクションの手順に従ってターゲットとの接続を作成し、将来のすべての接続でアプリケーションを root 権限で実行します。

TCF を使用するには

TCF を使用するには、MRAA/UPM を使用しているかどうかに関係なく、次の操作が必要です。

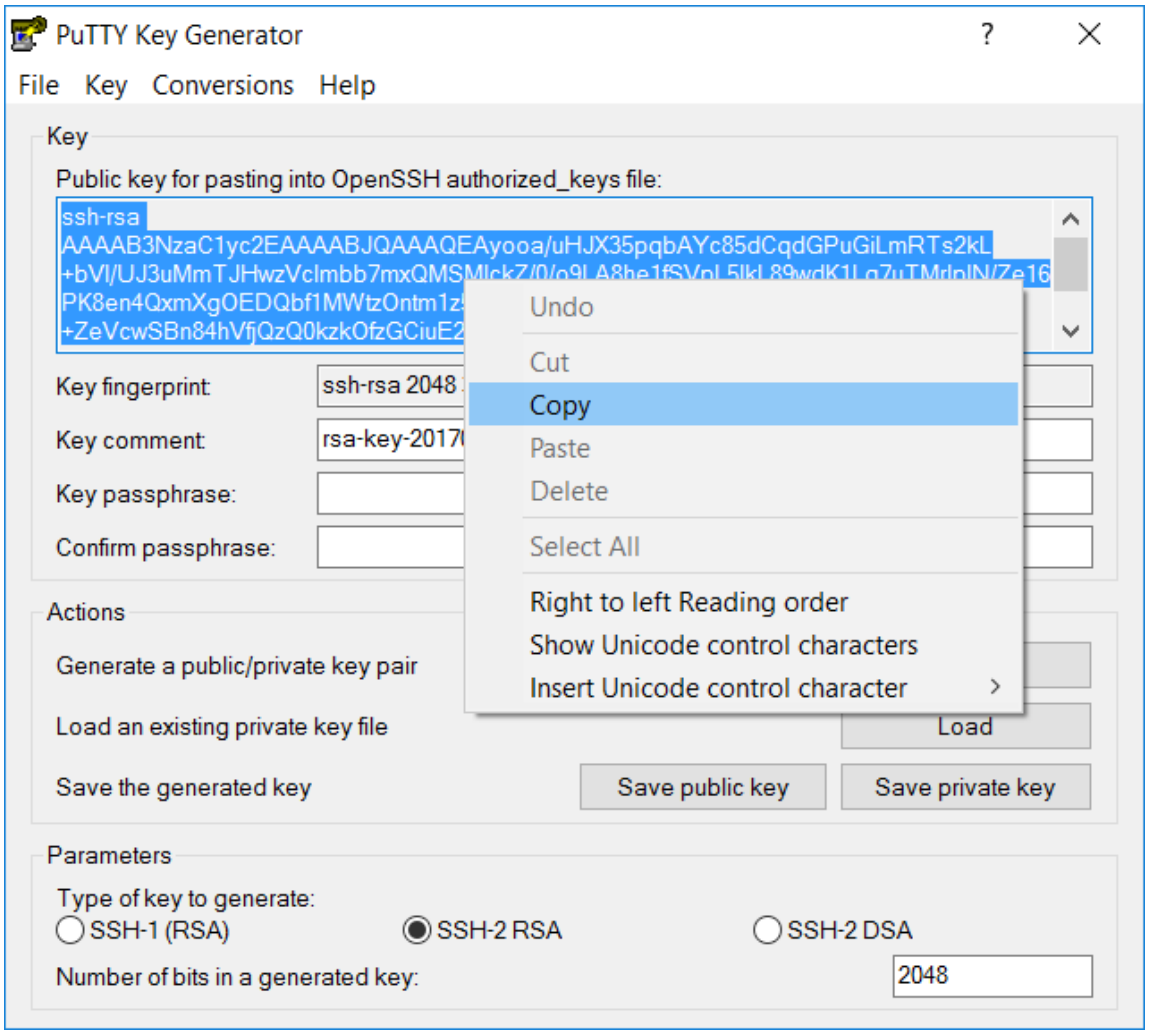
1. ホストとターゲットシステムの両方をネットワークに接続します。
2. ターゲット上に OpenSSH サーバーをインストールします。
   `sudo apt install openssh-server`

Linux* または macOS* ホストの場合

1. パスフレーズなしで鍵を作成します。
   `ssh-keygen -t rsa -f ~/.ssh/id_rsa_tcf`
   パスフレーズの設定を確認するメッセージが表示されたら **Enter** キーを押します。
2. 公開鍵をターゲットにコピーします。
   `scp ~/.ssh/id_rsa_tcf.pub {username}@{target_host}:`
3. ターゲットシステムに接続します。
   `ssh {username}@{target_host}`
4. ターゲットシステムで `sudo su -` を実行します。
5. `~/.ssh` ディレクトリーを作成します (存在しない場合)。
   `mkdir ~/.ssh`
6. `~/.ssh` ディレクトリーのアクセス権限を 700 に変更します。
   `chmod 700 ~/.ssh`
7. 公開鍵を `~/.ssh/authorized_keys` ファイルに追加します。
   `cat ~{username}/id_rsa_tcf.pub >> ~/.ssh/authorized_keys`
8. 次に、TCF 接続を設定します。「**TCF 接続の設定**」セクションの説明を参照してください。

Windows* ホストの場合

1. PuTTY SSH クライアント (英語) をインストールします。
2. `puttygen.exe` ファイルを使用して **PuTTY Key Generator** を起動します。
3. **[Generate]** ボタンをクリックして (デフォルト設定で) 鍵を生成します。
4. 秘密鍵をローカルドライブに保存します。
   `C:¥Users¥{username}¥Documents¥linux_private_key.ppk`
   PuTTY Key Generator は開いたままにします。
5. PuTTY を実行してターゲットシステムに接続します。
6. ターゲットシステムで `sudo su -` を実行します。
7. `~/.ssh` ディレクトリーを作成します (存在しない場合)。
   `mkdir ~/.ssh`
8. `~/.ssh` ディレクトリーのアクセス権限を 700 に変更します。
   `chmod 700 ~/.ssh`
9. テキストエディターで `~/.ssh/authorized_keys` ファイルを作成して開きます。
   `pico ~/.ssh/authorized_keys`
10. **PuTTY Key Generator** から公開鍵をコピーしてエディターに貼り付けます。



11. ファイルを保存してエディターを終了します。
12. 次に、TCF 接続を設定します。「**TCF 接続の設定**」セクションの説明を参照してください。

TCF 接続の設定

次の手順に従って、SSH 鍵を使用して TCF 接続を設定します。

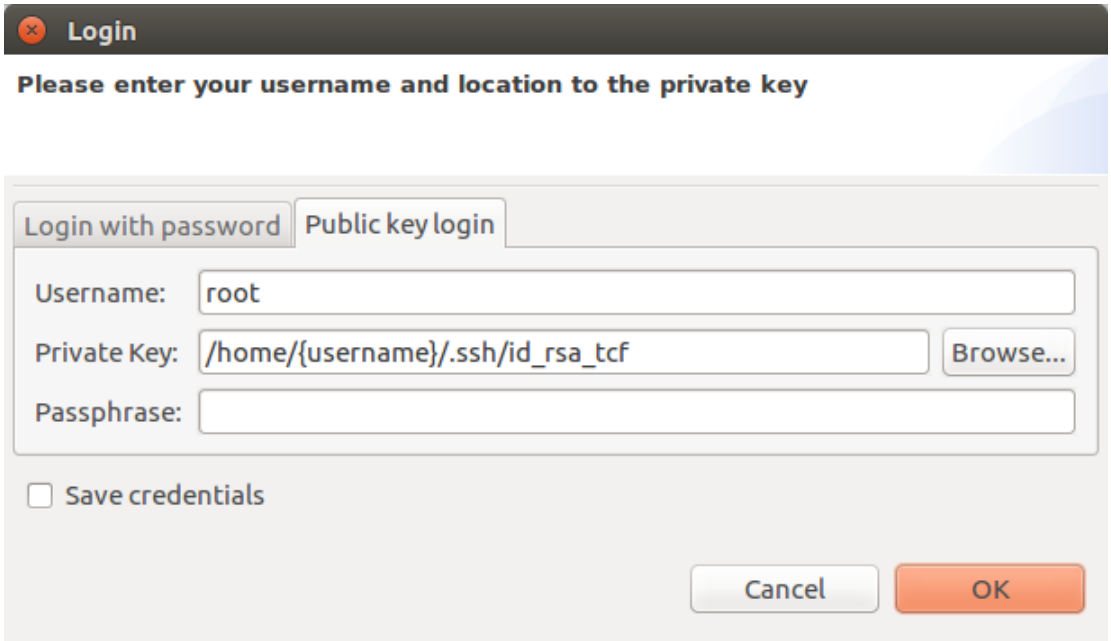
1. ツールバーで、**[New Connection]** をクリックします。
2. **[Connection for building in a container and running on Linux]** を選択して、**[Next]** をクリックします。
3. **[New Connection]** ウィンドウで、接続名と IP アドレスを入力します。**[Finish]** をクリックします。
4. **[Login]** ウィンドウで、**[Public key login]** タブをクリックします。
5. Username に、**root** と入力します。
6. 秘密鍵を参照します。

   Windows* の場合: 上記の「Windows* ホストの場合」セクションのステップ 4 で選択したパス、例えば `C:¥Users¥{username}¥Documents¥linux_private_key.ppk` を使用します。

   Linux* および macOS* の場合: パスは `/home/{username}/.ssh/id_rsa_tcf` です。

   注

   Linux* Ubuntu* で `.ssh` ディレクトリーが表示されていない場合は、**Ctrl + H** キーを押してディレクトリーを表示します。



注意

この方法を使用してターゲットに接続すると、インテル® System Studio は TCF エージェントをターゲットにコピーします。TCF エージェントは、ローカル・ネットワークのコンピューターからの接続を待ち受けます。ターゲットデバイスで TCF エージェントを開始するには、[IDE ターゲット接続] ダイアログで認証情報を設定する必要があります。ただし、TCF エージェントを開始した後はアクションの実行に認証は必要ありません。そのため、ターゲットデバイスへネットワーク・アクセス可能なすべてのユーザーは、ターゲットデバイスに接続して TCF エージェントを使用し、ターゲットデバイス上で任意のコードを実行することができます。エージェントをこのセクションの説明のように root で開始した場合、デバイスへネットワーク・アクセス可能なすべてのユーザーは、デバイス上で任意のコードを認証なしで実行することができます。

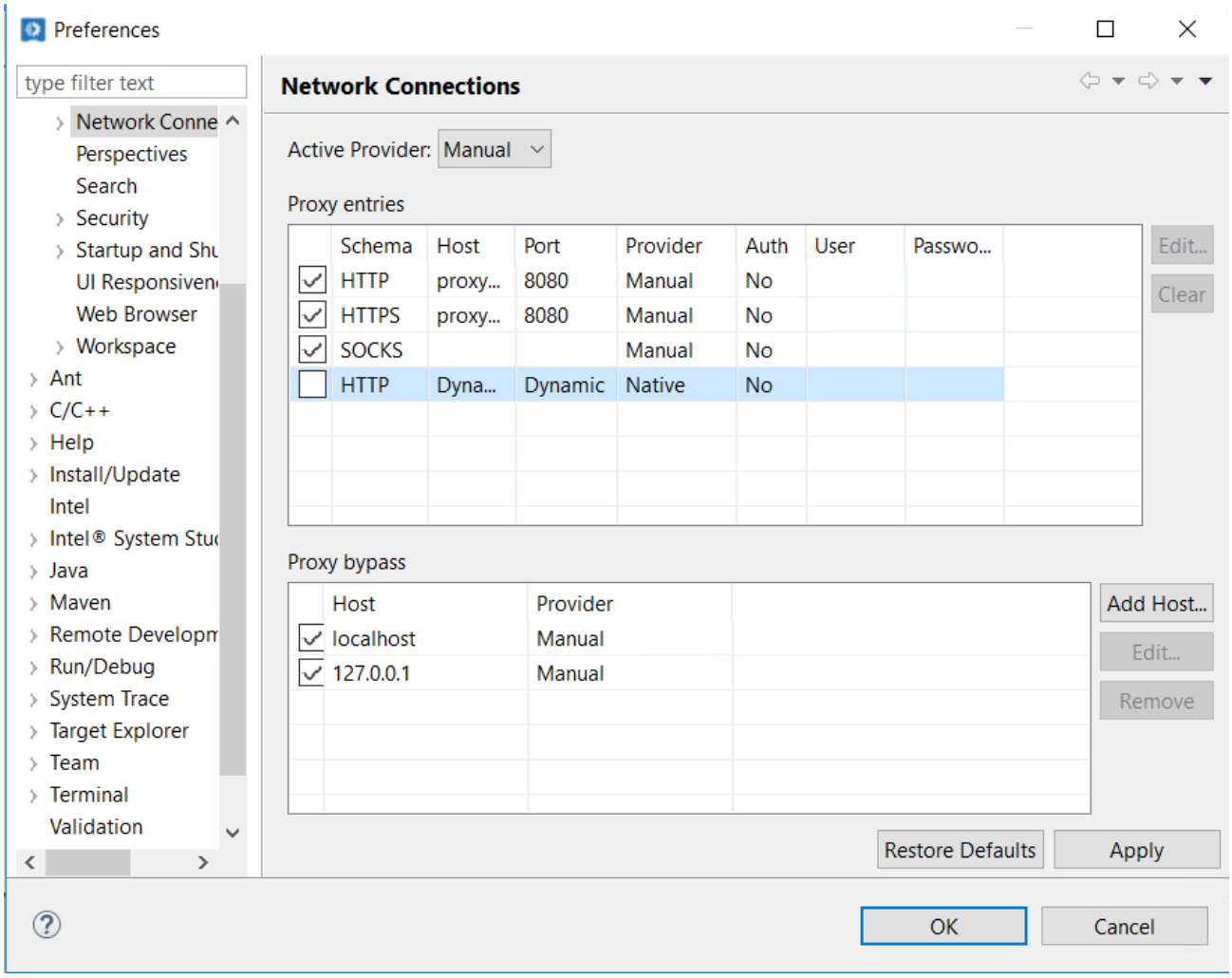
ネットワーク・プロキシー設定のセットアップ

次の手順に従って、インテル® System Studio のネットワーク・プロキシー設定をセットアップします。オープン・ネットワークを使用している場合は、次の手順はスキップしてください。

1. インテル® System Studio を開始します。
2. **[Window (ウィンドウ)] > [Preferences (設定)] > [General (全般)] > [Network Connections (ネットワーク接続)]** を選択して [Network Connections (ネットワーク接続)] ページを開きます。
3. **[Active Provider (アクティブ・プロバイダー)]** ドロップダウン・リストから、**[Manual (手動)]** を選択します。
4. 次の図のように、ネットワークのプロキシー情報を設定します。

   注

   Eclipse* で必要なのは、HTTP、HTTPS および Proxy bypass 設定のみです。ほとんどのネットワークでは、SOCKS プロキシーの値は空白です。
5. **[OK]** をクリックします。



Linux* ターゲット向けクロスコンパイル・プロジェクトの作成

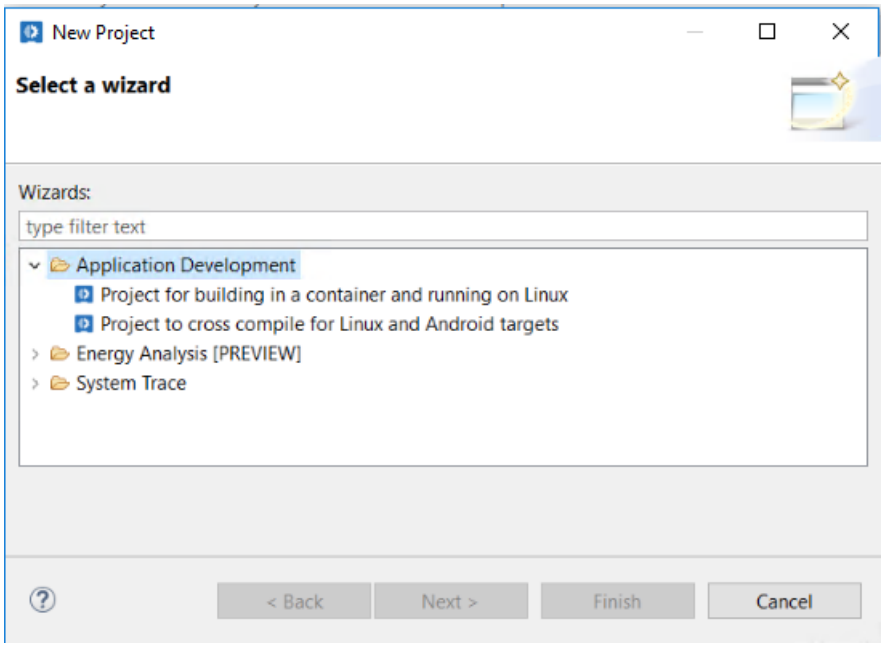
このセクションでは、インテル® C++ コンパイラーとクロスコンパイル・ワークフローを使用して Windows* ホスト /Linux* ターゲットで実行するアプリケーションを作成する方法を説明します。

注

Android* ターゲット向けの開発に関する情報は、「Android* Studio でインテル® C++ コンパイラーを使用したネイティブ Android* アプリケーションのビルド」(英語) を参照してください。

ステップ 1: 新規プロジェクトの作成

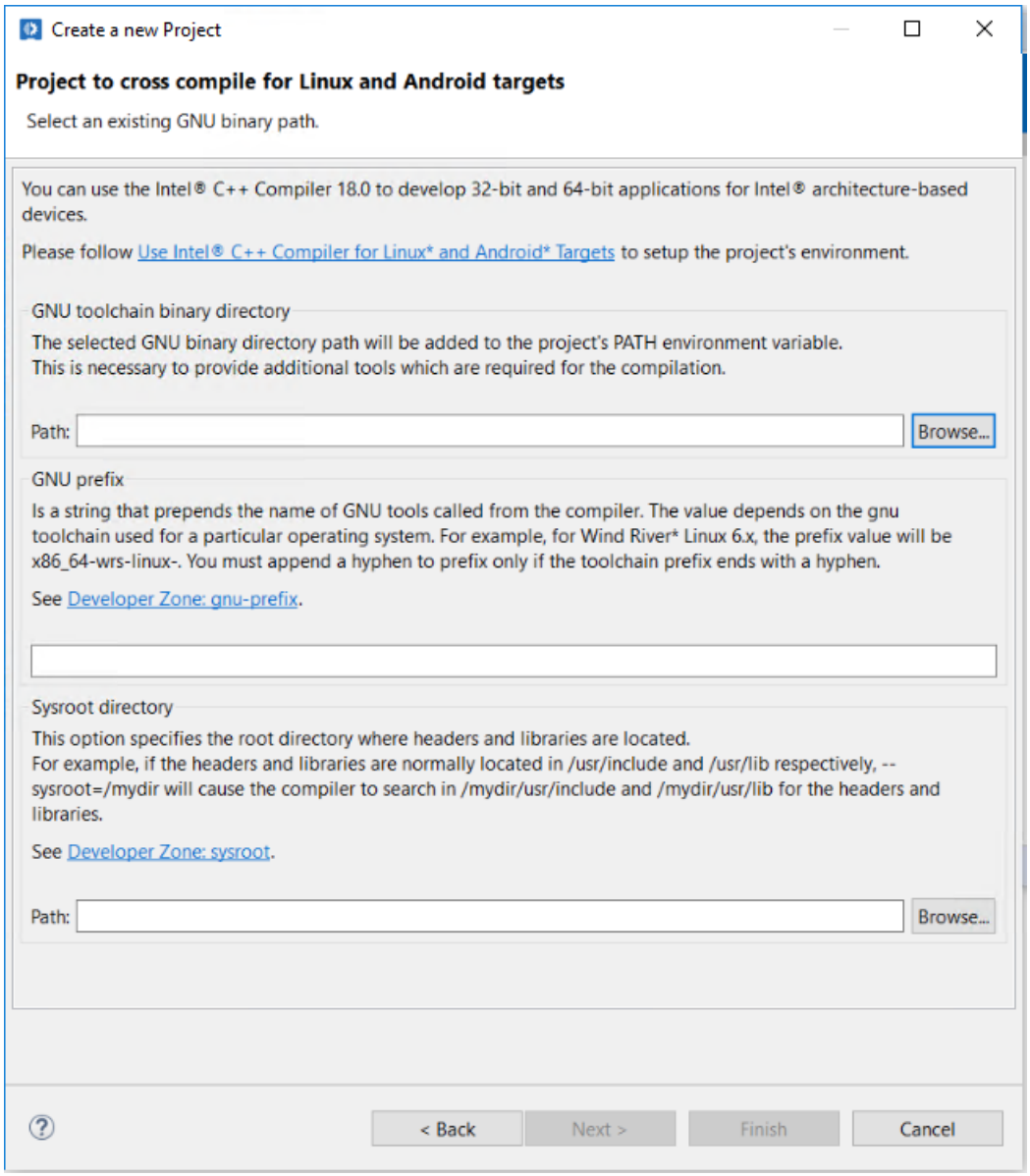
1. Eclipse* で、**[File]** > **[New]** > **[Project]** をクリックします。



2. **[Application Development]** を展開して **[Project to cross compile for Linux and Android targets]** を選択します。
3. **[Next]** をクリックします。[Create a New Project] 画面が表示されます。

ステップ 2: クロスビルド・オプションのセットアップ

1. GNU* バイナリー・ディレクトリーのパスを選択します。選択したパスはプロジェクトの PATH 環境変数に追加されます。
2. [GNU prefix] には、コンパイラーから呼び出される GNU* ツールの名前の前に追加する文字列を指定します。詳細は、「gnu-prefix」(英語) の説明を参照してください。
3. [Sysroot directory] には、ヘッダーとライブラリーのターゲット root ディレクトリーを指定します。詳細は、「sysroot」(英語) の説明を参照してください。
4. **[Next]** をクリックします。



5. プロジェクトの作成を完了します。

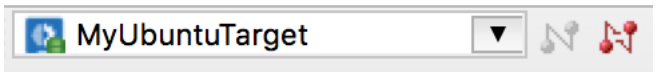
ステップ 3: ターゲット接続の選択

プロジェクトを実行またはデバッグするときにプロジェクトが使用する接続はターゲット接続と呼ばれます。インテル® System Studio のプロジェクトは現在選択されているターゲット接続を使用します。

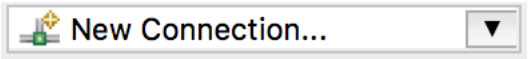
注意

この方法を使用してターゲットに接続すると、インテル® System Studio は TCF エージェントをターゲットにコピーします。TCF エージェントは、ローカル・ネットワークのコンピューターからの接続を待ち受けます。ターゲットデバイスで TCF エージェントを開始するには、[IDE ターゲット接続] ダイアログで認証情報を設定する必要があります。ただし、TCF エージェントを開始した後はアクションの実行に認証は必要ありません。そのため、ターゲットデバイスへネットワーク・アクセス可能なすべてのユーザーは、ターゲットデバイスに接続して TCF エージェントを使用し、ターゲットデバイス上で任意のコードを実行することができます。

1. 既存のターゲット接続を選択します。初めてプロジェクトを作成する場合は、新しい接続を作成します。
   a. **既存の接続を選択する場合:** 接続ドロップダウン・リストの横にある矢印をクリックしてターゲット接続を選択します。



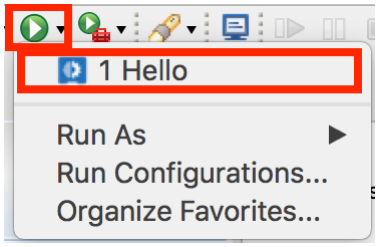
b. **新しい接続を作成する場合**: **[New Connection]** をクリックします。



新しい接続のセットアップを完了します。

ステップ 4. プロジェクトの実行またはデバッグ

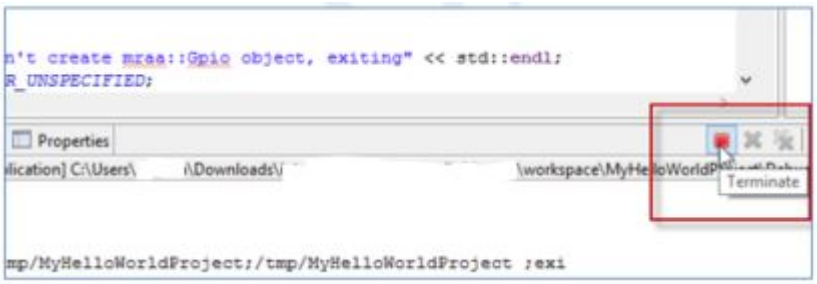
1. プロジェクトを実行するには、**[Run]** ボタンをクリックしてドロップダウン・リストからプロジェクトの名前を選択します。



プロジェクトをデバッグするには、 **[Debug]** ボタンをクリックしてドロップダウン・リストからプロジェクトの名前を選択します。

2. ホストの認証に関する警告メッセージが表示された場合は、**[Yes (はい)]** をクリックしてプロジェクトをアップロードして実行します。

3. プロジェクトが実行されます。
プロジェクトが自動的に終了しない場合、[Console] パネルに表示されている赤の **[Terminate]** アイコンをクリックします。



詳細は、次の情報を参照してください。

- インテル® C++ コンパイラー 18.0 デベロッパー・ガイドおよびリファレンス (英語)
- インテル® System Studio 2018 向けインテル® C++ コンパイラー 18.0 デベロッパー・ガイドおよびリファレンス補足 (英語)

関連資料

資料	説明
コードサンプルとチュートリアル (英語)	新しいユーザー向けに基本的な製品の機能を説明する、ダウンロード可能なサンプルコードとチュートリアル。サンプルコードは一般的なシナリオを例示し、対応するチュートリアルはビルドツールを使用してコードを作成および最適化する方法を示します。
インテル® C++ コンパイラー入門ガイド (英語)	インテル® C++ コンパイラーの基本情報および追加のヘルプへのリンク
インテル® MKL 入門ガイド (英語)	インテル® MKL の基本情報および追加のヘルプへのリンク
インテル® TBB 入門ガイド (英語)	インテル® TBB の基本情報および追加のヘルプへのリンク
インテル® IPP 入門ガイド (英語)	インテル® IPP の基本情報および追加のヘルプへのリンク
インテル® DAAL 入門ガイド (英語)	インテル® DAAL の初歩的な情報および追加のヘルプへのリンク

インテル® System Studio ログファイル	インテル® System Studio インストーラーは、ログファイルを `AppData¥Local¥Temp¥Intel` に書き込みます。ログファイルの名前は `intel.pset` で始まり、タイムスタンプで終了します。拡張子は `.log` です。
	例:
	`C:¥Users¥auser¥AppData¥Local¥Temp¥Intel¥intel.pset.root.auser_MOBL1_Thu_10_26_2017__02.28_PM.log`
	インテル® System Studio は、ログファイルのペアを `AppData¥Local¥Temp` のサブディレクトリーに書き込みます。サブディレクトリー名は `iss_env_$USER_$TIMESTAMP` 形式です。`session.log` ファイルには環境設定が含まれます。`eclipse.log` ファイルにはセッションログが含まれます。
	例:
	`C:¥Users¥auser¥AppData¥Local¥Temp¥iss_auser_20171026164410¥eclipse.log` および
	`C:¥Users¥ auser¥AppData¥Local¥Temp¥iss_auser_20171026164410¥session.log`

資料	説明
その他のドキュメント	最新の情報は、オンライン・ドキュメント (英語) を参照してください。 インテル® System Studio のすべてのドキュメントを含む ZIP ファイルを https://software.intel.com/en-us/articles/download-documentation-intel-system-studio-current-previous (英語) からダウンロードしてオフラインで利用できます。
インテル® System Studio 製品ページ	サポートおよび最新のドキュメントに関する情報

インテル® System Studio 2018 for Windows* ホスト/Linux* ターゲットを使用したプロファイルおよび解析

解析コンポーネントの概要

インテル® System Studio Professional Edition および Ultimate Edition には、特定のユースケースに対応したさまざまな解析ツールが用意されています。パフォーマンス問題を解決するには、(複数の) 解析ツールを使用して問題を正確に特定し、システムを最適化します。

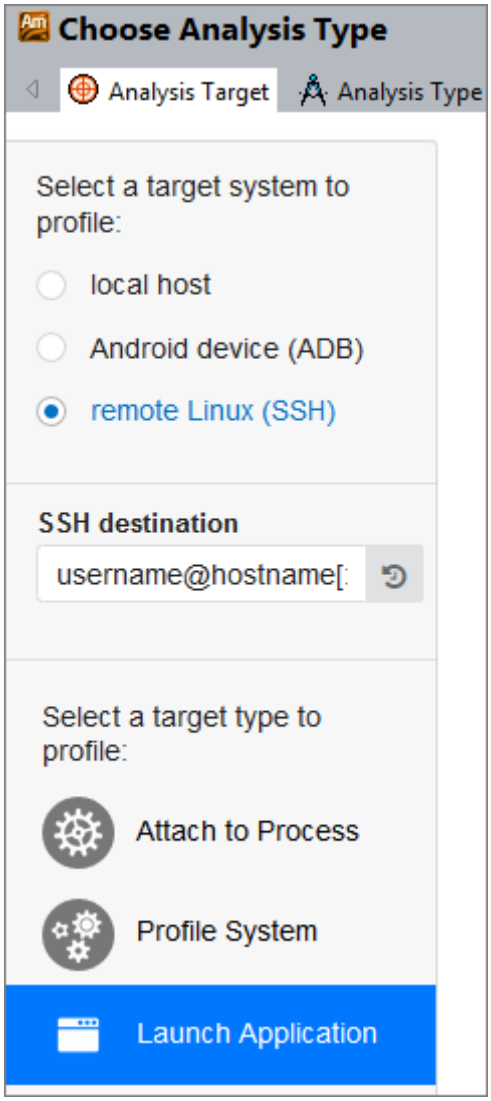
- インテル® VTune™ Amplifier: ローカルまたはリモートシステムでのさまざまなユースケースに対応する解析タイプを提供します。コードの hotspot の検証、メモリー問題の特定、複数の CPU コアにわたるスレッド化の最適化など。
- インテル® Inspector: コードのメモリーエラーとスレッドエラーをデバッグします。
- 電力解析とインテル® SoC Watch (英語): 電力消費の解析および電力を浪費しているシステム動作の識別に使用できるメトリックを収集します。結果は .CSV ファイル、インテル® System Studio (Linux* ターゲットのみ) またはインテル® VTune™ Amplifier で表示できます。
- インテル® Graphics Performance Analyzers: グラフィックスを多用するアプリケーションの解析と最適化を行います。

インテル® VTune™ Amplifier

注

準備: リモート Linux* システムで解析を実行する場合は、ターゲットシステムへの SSH 接続 (英語) を設定します。

1. Eclipse* ウィンドウで、**[Intel System Studio]** > **[VTune Amplifier]** > **[Launch VTune Amplifier]** を選択します。
2. インテル® VTune™ Amplifier ウィンドウで、new project (新規プロジェクト) をクリックし、名前と場所を指定して、**[Create Project (プロジェクトの作成)]** をクリックします。
3. **[Analysis Target (解析ターゲット)]** タブで、ターゲットシステムと解析ターゲットタイプを選択します。ターゲットシステムとして **[remote Linux (SSH)]** を選択した場合は、**[SSH destination]** を設定します。



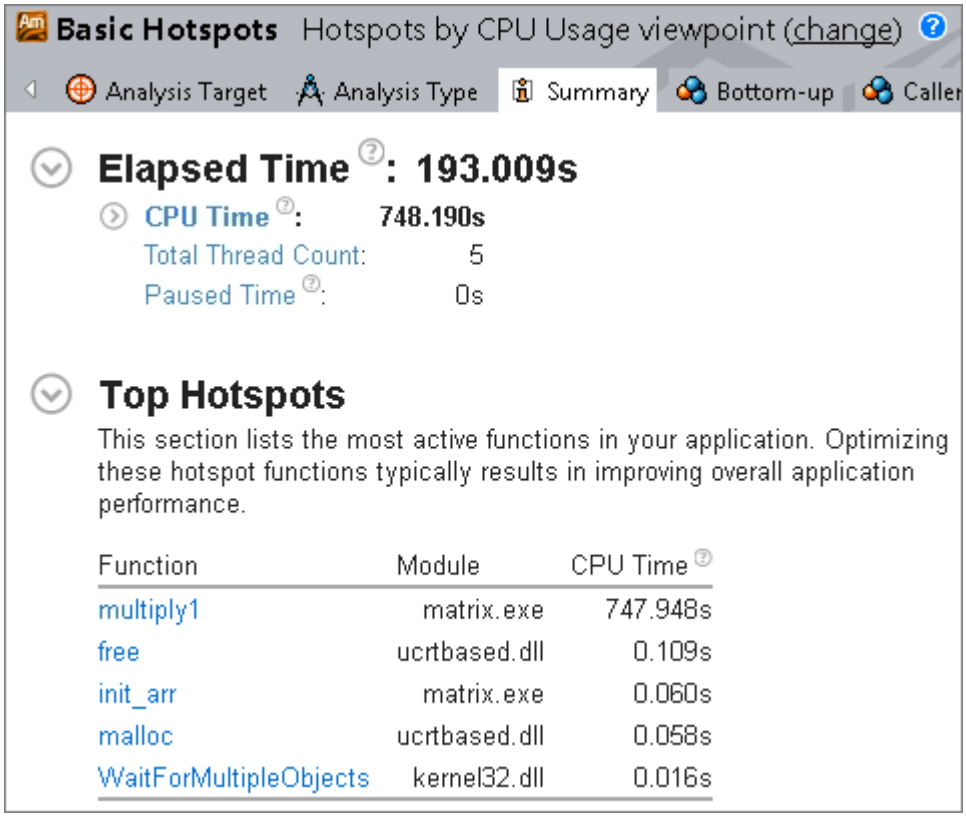
4. **[Analysis Target (解析ターゲット)]** タブをクリックし、解析タイプを選択して、解析を実行します。

   ヒント

   ユースケースに最適な解析タイプを素早く選択するには、インテル® VTune™ Amplifier の [Welcome (ようこそ)] 画面から利用可能な [Find Your Analysis (解析タイプの特定)] ガイドを使用します。次の最適化を行います。

   - **hotspot**: Basic Hotspots (基本 hostpot) 解析および Advanced Hotspots (高度な hotspot) 解析
   - **並列処理**: Concurrency (コンカレンシー) 解析、Locks and Waits (ロックと待機) 解析、および HPC Performance Characterization (HPC パフォーマンス特性) 解析
   - **マイクロアーキテクチャー**の問題: General Exploration (全般) 解析および Memory Access (メモリーアクセス) 解析

5. パフォーマンス・サマリーでハイライトされた問題に注目して、最適化の手順に従います。



詳細は、以下の資料を参照してください。

資料	説明
入門ガイド (英語)	インテル® VTune™ Amplifier を実行するための基本的な手順を説明します。
オンラインヘルプ	インテル® VTune™ Amplifier の主要ドキュメントです。
チュートリアル (英語)	インテル® VTune™ Amplifier の基本的な機能の使用方法を説明します。新しいユーザー向けに簡単なサンプルを使用して基本的な操作を紹介します。インテル® VTune™ Amplifier ヘルプを参照する前にチュートリアルをご覧になることを推奨します。
クックブック	インテル® VTune™ Amplifier の解析タイプを利用した最も一般的なパフォーマンス問題の識別と解決のレシピを含むパフォーマンス解析クックブックです。

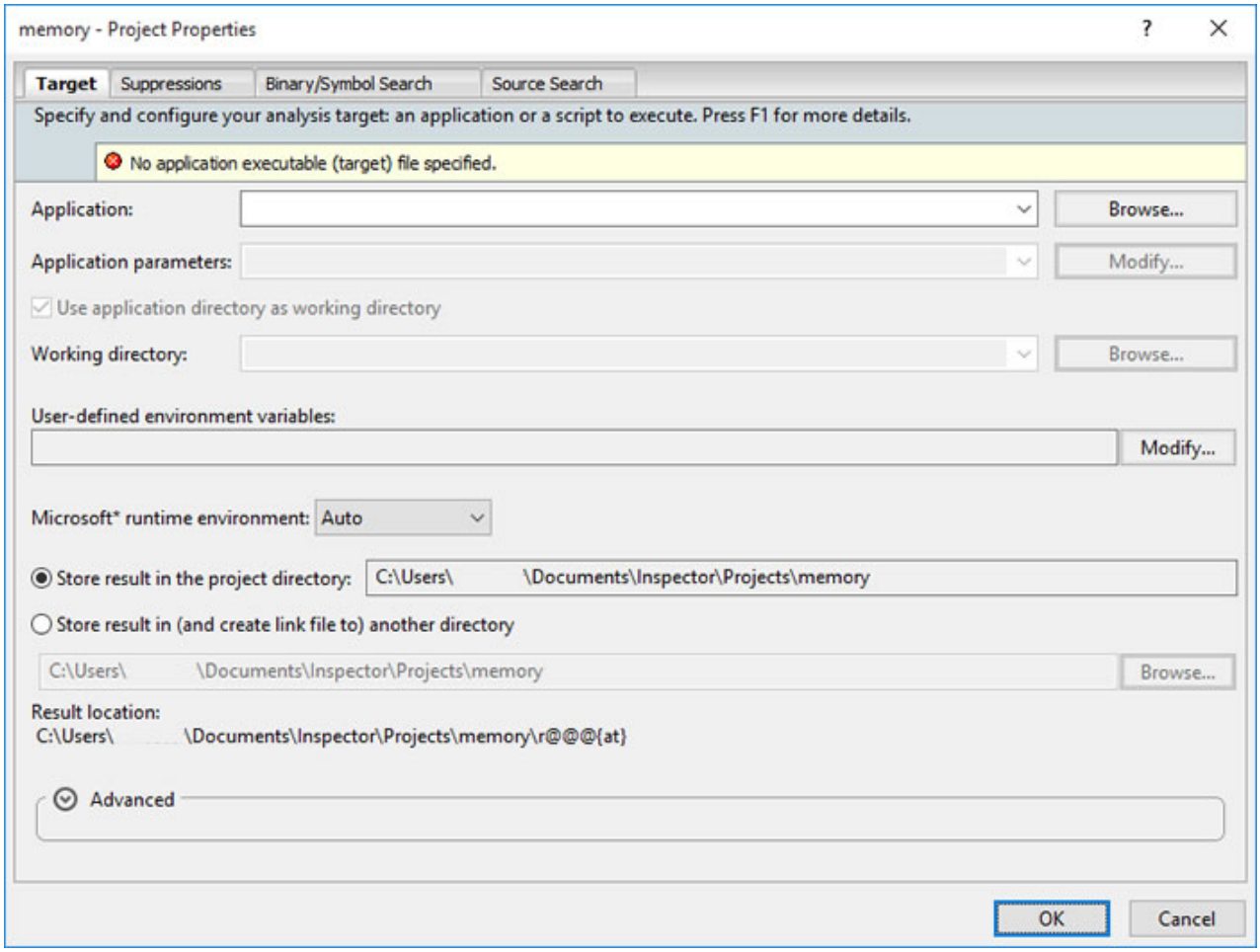
インテル® Inspector

インテル® Inspector はホストシステムでのみ利用できます。リモート・ターゲット・システムのアプリケーションは解析できません。代わりに、ホストシステムで事前にビルドされたアプリケーションを使用して解析を実行します。

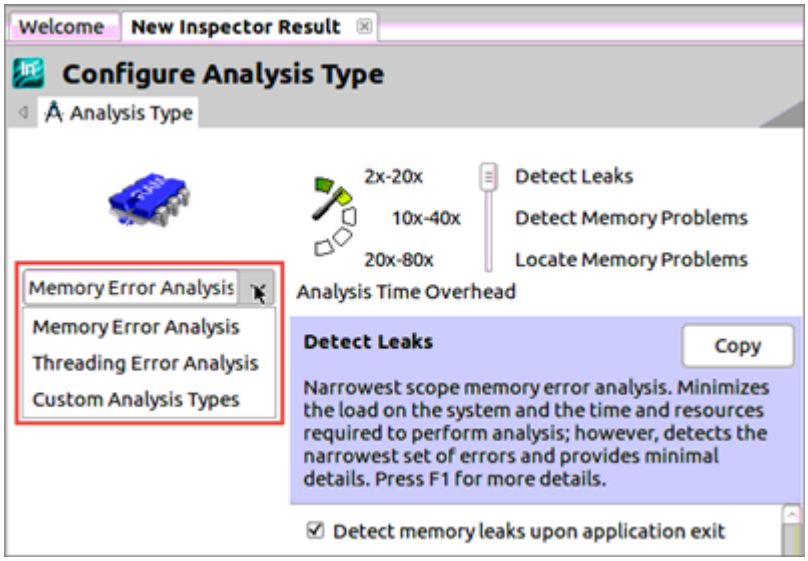
1. Eclipse* ウィンドウで、**[Intel System Studio]** > **[Inspector]** > **[Launch Inspector]** を選択します。
2. インテル® Inspector ウィンドウで、**[File (ファイル)]** > **[New (新規)]** > **[Project (プロジェクト)]** を選択し、プロジェクト名を指定して、**[Create a Project (プロジェクトの作成)]** をクリックします。
3. 事前にビルドされたアプリケーションを参照して **[OK]** をクリックします。

注

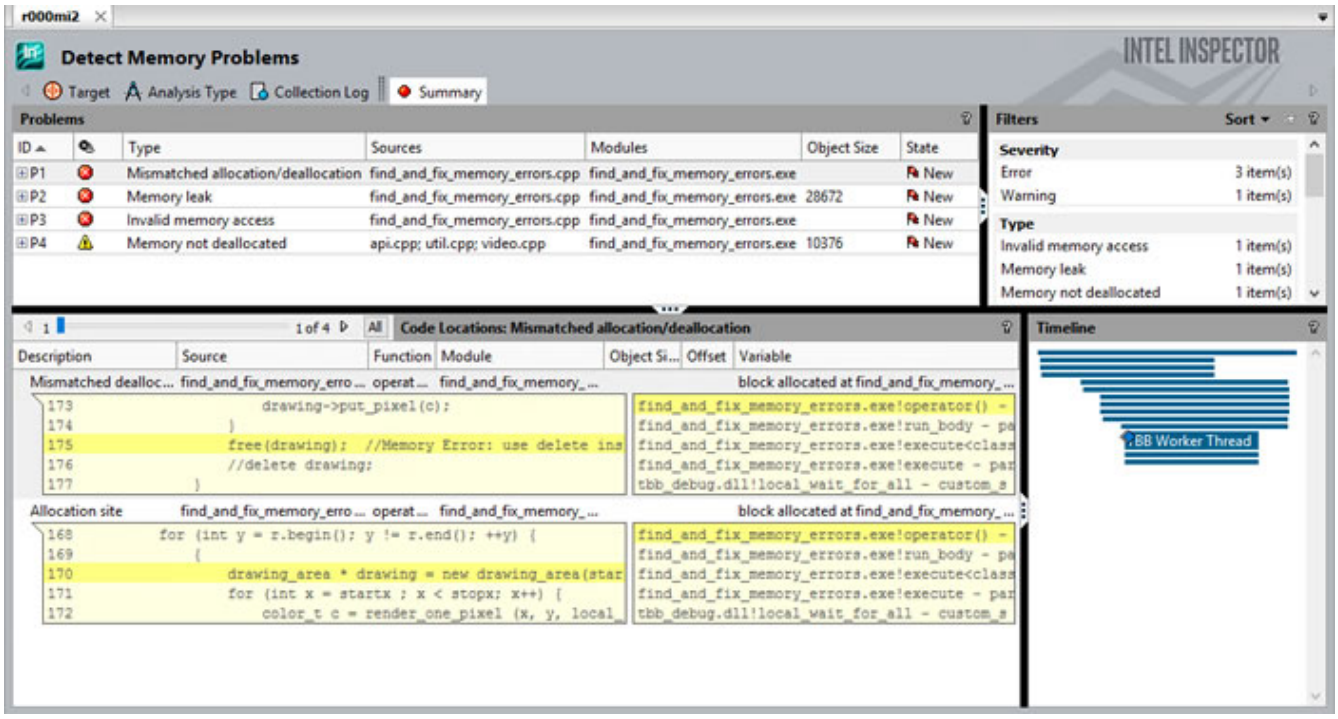
特別なコンパイラーやビルドは必要ありません。通常のデバッグビルドまたはプロダクション・ビルドを利用できます。



4. **[File (ファイル)]** > **[New (新規)]** > **[Analysis (解析)]** を選択し、ドロップダウンから解析タイプを選択して、解析を実行します。



5. **[Problems (問題)]** ペインの優先順位の付いた作業リストに注目します。問題をクリックすると、下のペインに詳細な情報が表示されます。問題をダブルクリックすると、ソースコードが表示されます。



詳細は、以下の資料を参照してください。

資料	説明
入門ガイド (英語)	インテル® Inspector を実行するための基本的な手順を説明します。
ヘルプ - Windows* (英語)	インテル® Inspector の主要ドキュメントです。
チュートリアル (英語)	インテル® Inspector の基本的な機能の使用方法を説明します。新しいユーザー向けに簡単なサンプルを使用して基本的なワークフローを紹介します。

電力解析

ワークフローに最も適した電力解析を選択します。

- 電力解析 Eclipse* プラグインを使用して、Linux* または Windows* ホストシステムからリモート Linux* ターゲットで解析を実行して結果を表示します。これはプレビュー機能です。詳細は、『電力解析のヘルプ』(英語) を参照してください。
- インテル® SoC Watch コマンドライン・ツールを実行して、ターゲットシステムで電力解析の結果のサマリーを素早く表示します。詳細は、「インテル® SoC Watch 入門ガイド」(英語) を参照してください。
- 結果をインテル® VTune™ Amplifier にインポートして、電力解析の結果を時間とともに表示します。詳細は、『電力解析のヘルプ』(英語) を参照してください。

詳細は、以下の資料を参照してください。

資料	説明
入門ガイド (英語)	インテル® SoC Watch コマンドライン・ツールを実行して電力解析を実行するための基本的な手順を説明します。
電力解析のヘルプ (英語)	インテル® SoC Watch、インテル® VTune™ Amplifier、およびインテル® System Studio を使用してローカルおよびリモートシステムで電力プロファイルを実行して解釈する方法を説明します。
インテル® SoC Watch ユーザーガイド Linux* および Android* (英語) | Windows* (英語)	インテル® SoC Watch で収集される各メトリックに関する詳細な情報を提供します。

インテル® Graphics Performance Analyzers

Graphics Monitor は、グラフィックスを多用する DirectX*、OpenGL*、および OpenGL* ES アプリケーションを最適化する 3 つの強力なアジャイルツールの出発点です。System Analyzer、Graphics Frame Analyzer、および Platform Analyzer が含まれています。

ステート・オーバーライドを試行して、パフォーマンス問題をグラフィックス・パイプラインの特定段階に関連付けます。試行結果に応じて、次の最適化の可能性を調べます。

- テクスチャー帯域幅の解析
- 複雑なジオメトリーの最適化
- ステートとシェーダーによる試行
- オーバードローの最小化

詳細は、以下の資料を参照してください。

資料	説明
入門ガイド (英語)	グラフィックス解析を実行するための基本的な手順を説明します。
インテル® Graphics Performance Analyzers のドキュメント (英語)	最新の情報は、オンライン・ドキュメント (英語) を参照してください。 インテル® System Studio のすべてのドキュメントを含む ZIP ファイルを https://software.intel.com/en-us/articles/download-documentation-intel-system-studio-current-previous (英語) からダウンロードしてオフラインで利用できます。

関連資料

資料	説明
コードサンプルとチュートリアル (英語)	新しいユーザー向けに基本的な製品の機能を説明する、ダウンロード可能なサンプルコードとチュートリアル。サンプルコードは一般的なコードの非効率性を例示し、対応するチュートリアルはパフォーマンス解析製品を使用して問題を解決する方法を示します。
その他のドキュメント	最新の情報は、オンライン・ドキュメント (英語) を参照してください。 インテル® System Studio のすべてのドキュメントを含む ZIP ファイルを https://software.intel.com/en-us/articles/download-documentation-intel-system-studio-current-previous (英語) からダウンロードしてオフラインで利用できます。
インテル® System Studio 製品ページ	サポートおよび最新のドキュメントに関する情報

インテル® System Studio 2018 for Windows* ホスト/Linux* ターゲットを使用したハードウェア・システム/プラットフォームの起動

デバッグ・コンポーネントの概要

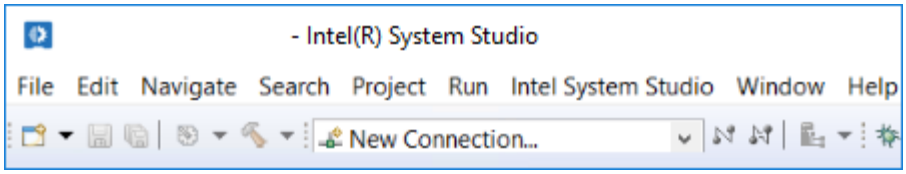
コンポーネント	説明
インテル® System Debugger — システムデバッグ	インテル® アーキテクチャー・ベースのシステム・ソフトウェアと組込みアプリケーションの JTAG ベースのデバッグを有効にします。
インテル® System Debugger — システムトレース	高度な SoC ワイドの命令とデータイベントのトレースを提供する、インテル® Trace Hub へアクセスできるようにします。

インテル® System Debugger — システムデバッグを使用した Simics* シミュレーション・ターゲットのデバッグ

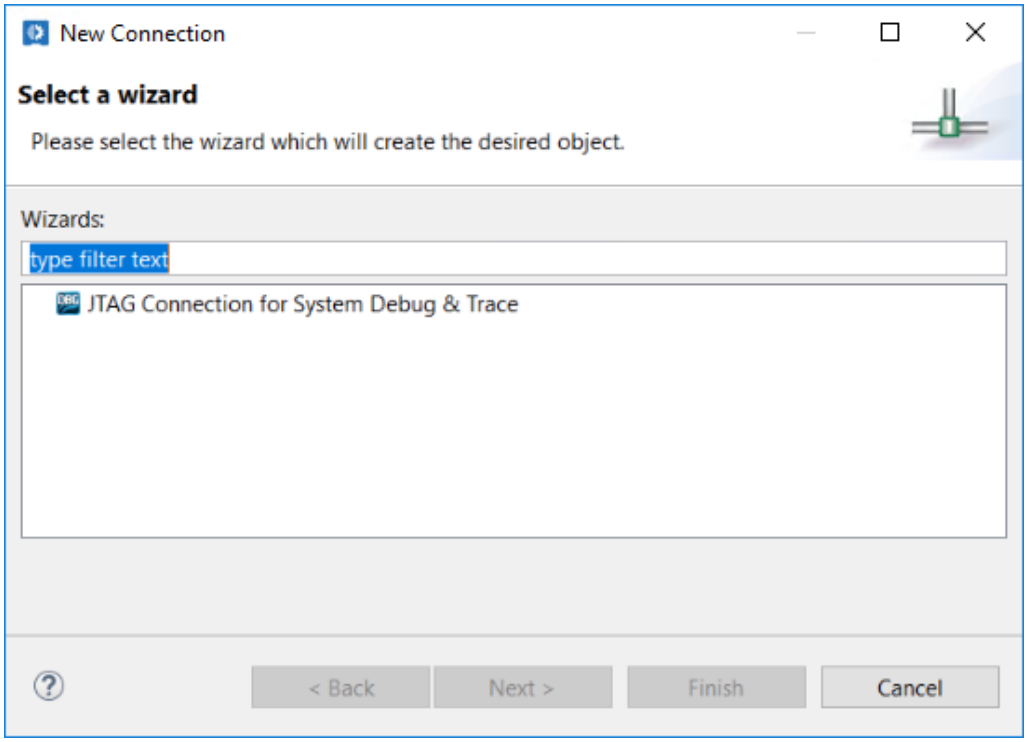
このチュートリアルは、Simics* シミュレーション・ターゲットを使用して、インテル® System Debugger — システムデバッグの基本的な機能を理解できるようにします。

ステップ 1: ターゲット接続の確立

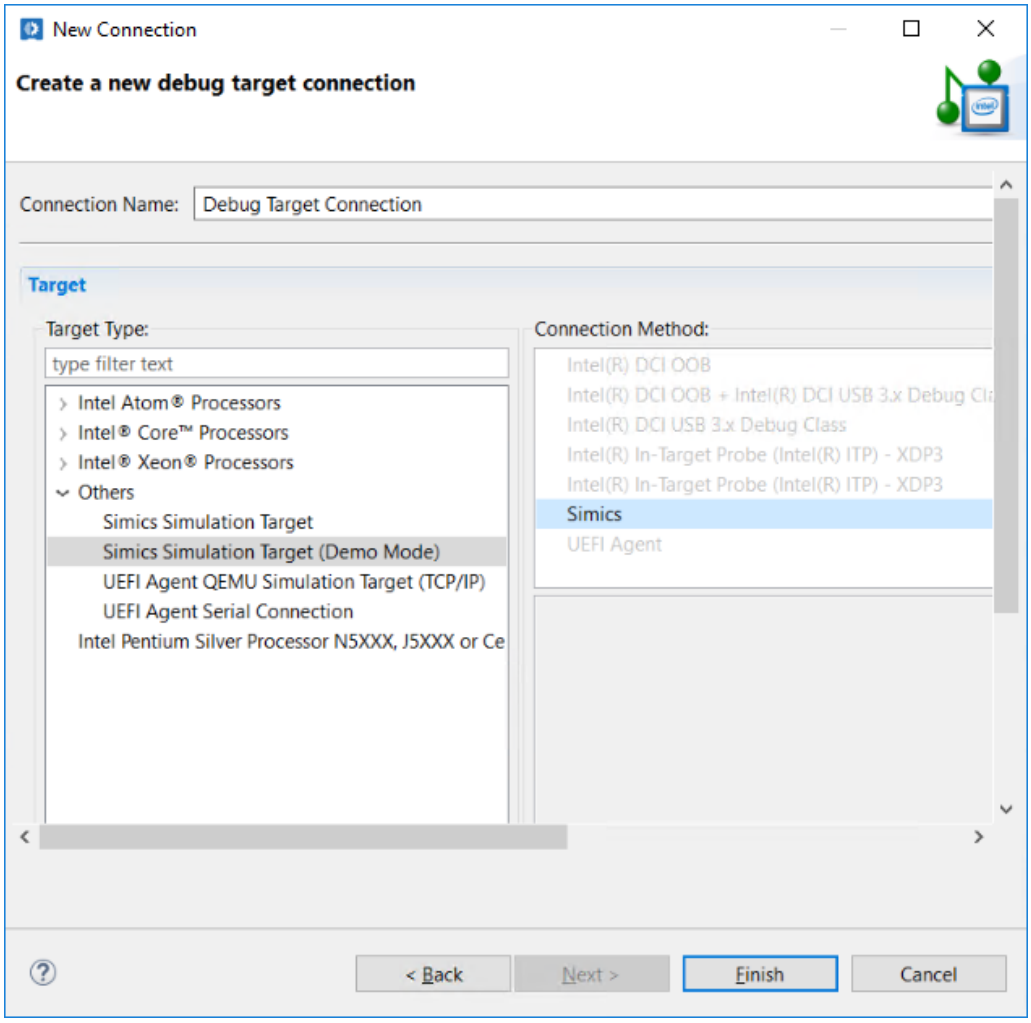
1. **[スタート]** メニューで **[Intel(R) System Studio (インテル(R) System Studio)]** をクリックしてインテル® System Studio IDE を起動します。あるいは、スクリプト `<install-dir>¥iss_ide_eclipse-launcher.bat` (`<install-dir>` はインテル® System Studio のインストール・ディレクトリー) を実行します。
2. インテル® System Studio IDE のメインツールバーで **[New Connection... (新規接続...)]** ドロップダウン・リストをクリックします。



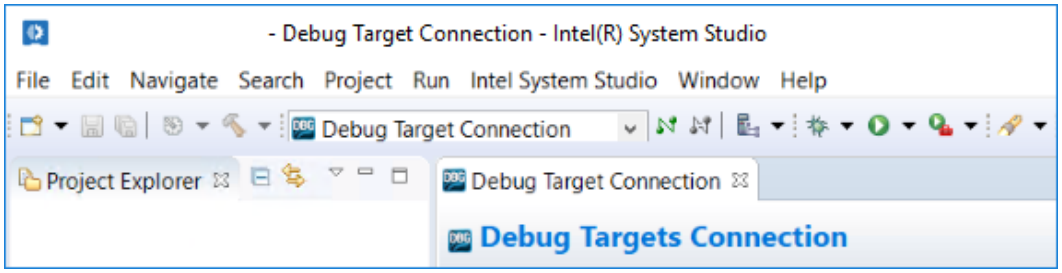
3. **[Wizards (ウィザード)]** リストで、**[JTAG Connection for System Debug & Trace (システムデバッグ/トレースの JTAG 接続)]** を選択して **[Next > (次へ >)]** をクリックします。



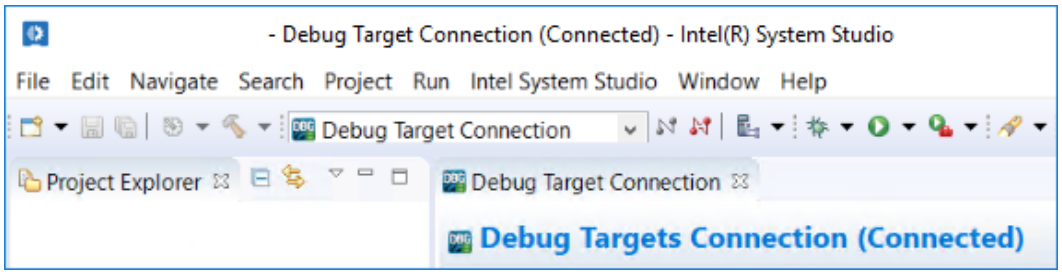
4. **[Connection Name (接続名)]** 入力フィールドに接続の名前を入力した後、**[Target Type (ターゲットタイプ)]** リストの **[Others (その他)]** エントリーを展開して **[Simics Simulation Target (Demo Mode) (Simics* シミュレーション・ターゲット (デモモード))]** を選択します。**[Connection Method (接続方法)]** リストで **[Simics (Simics*)]** を選択して **[Finish (完了)]** をクリックします。



5. ターゲット選択ドロップダウン・リストの横にある緑の **[Connect (接続)]** ボタンを押します。

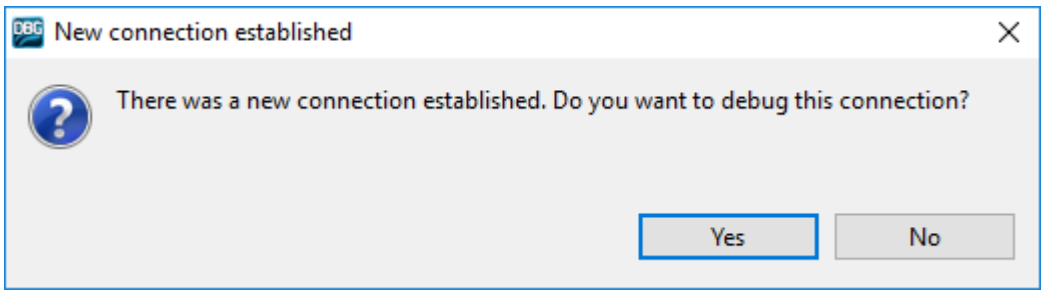


6. 接続が確立すると、**[Disconnect (切断)]** ボタンが有効になり、インテル® System Studio IDE のタイトルバーと [Debug Targets Connection (デバッグターゲット接続)] タブに **(Connected)** と表示されます。



ステップ 2: デバッグセッションの開始

1. インテル® System Studio IDE の **[Launch Intel System Debugger (Debug) (インテル(R) System Debugger の起動 (デバッグ))]** ボタンをクリックしてインテル® System Debugger を起動します。
2. **[New connection established (新規接続確立)]** ダイアログボックスで **[Yes (はい)]** をクリックします。



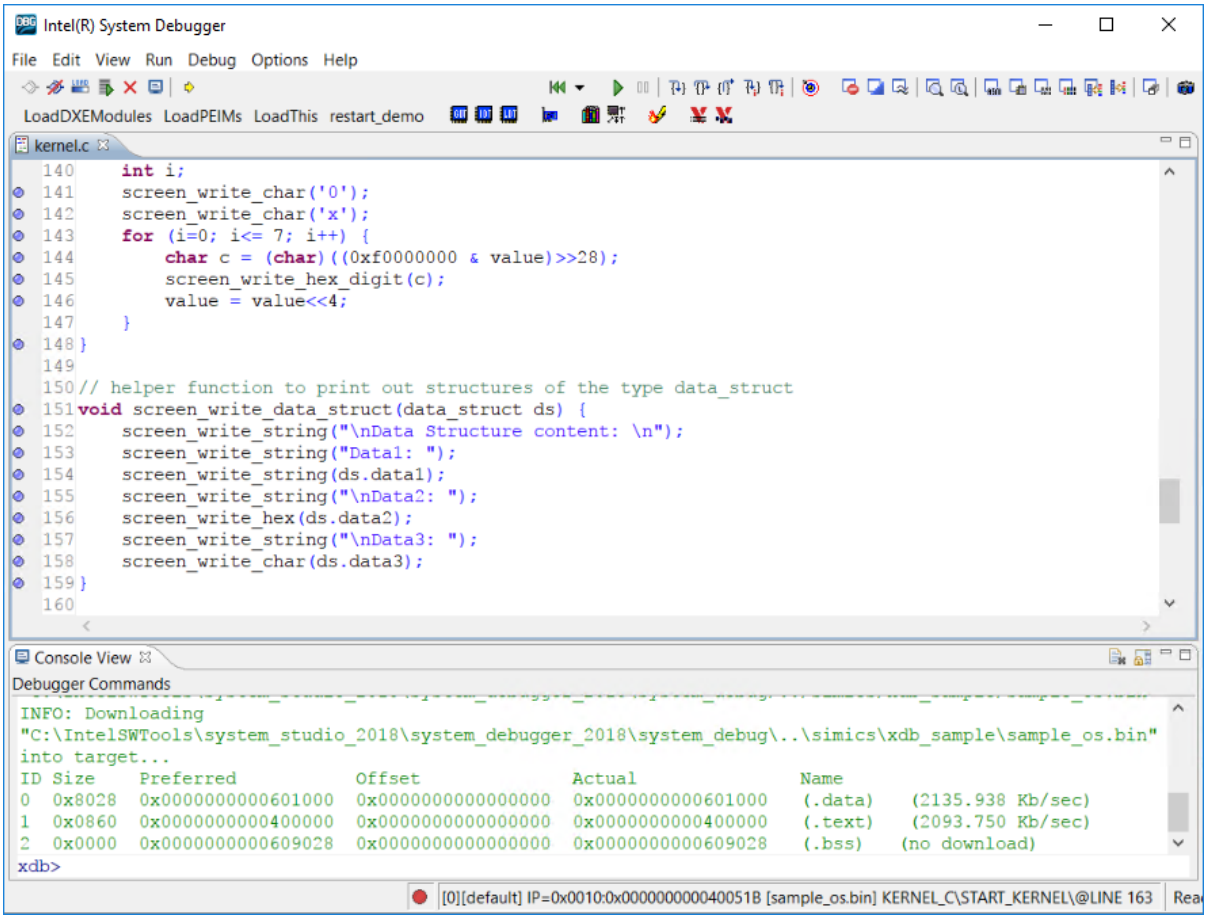
3. 初めてデモを開始した場合は、フィードバックに関するダイアログボックスが表示されます。項目を選択して **[OK]** をクリックします。

   注

   Windows* システムでは、セキュリティー設定に応じて、[Windows セキュリティの重要な警告] ダイアログボックスが表示され、デモ・アプリケーションの機能のいくつかが Windows* ファイアウォールでブロックされていると通知される場合があります。この通知が表示された場合は、適切な設定を選択して **[アクセスを許可する]** をクリックします。設定が保存され、以降のセッションでアプリケーションを起動したときにダイアログボックスが表示されなくなります。

デモ・アプリケーションの起動中、シミュレーション・プロセスを開始するシェルおよびシミュレートされたグラフィックスとシリアルコンソールが出力する、いくつかのシミュレーション関連のウィンドウが表示されます。このチュートリアルでは、これらのウィンドウは無視してかまいません。

スタートアップが完了すると、インテル® System Debugger がデフォルトビューで表示されます。デバッガーはプログラムコードをターゲット (シミュレーション・エンジン) にダウンロードして、デモ・アプリケーションの実行をエントリーポイントで停止します。



[Source (ソース)] ビューと **[Console (コンソール)]** ビューに、アクティブなソースコードとデバッガーの状況が表示されます。

[Console (コンソール)] ビューはコマンドラインのように動作します。このビューにインテル® System Debugger のコマンドを入力できます。デバッグの状況に関する詳細な情報を提供する、情報、警告またはエラーメッセージも表示されます。

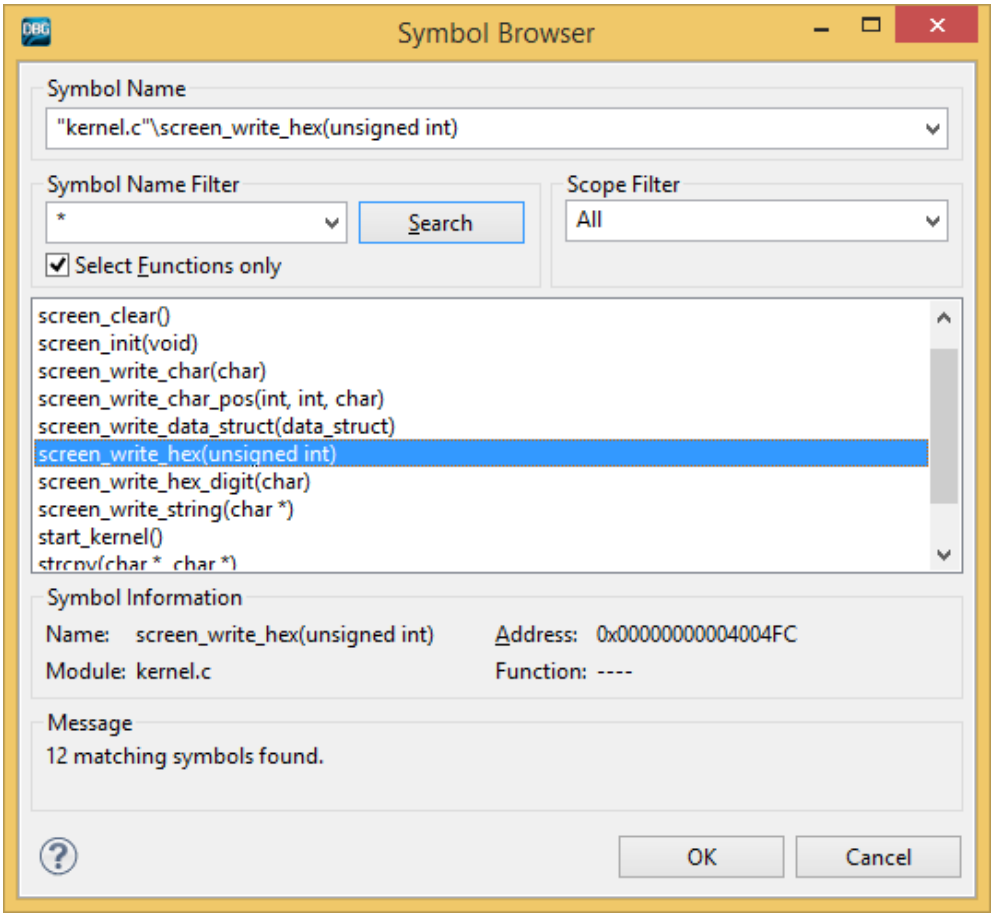
ヒント

問題が発生したら、**[Console (コンソール)]** ビューの内容を確認します。コンソールはデフォルトでは表示されない場合があります。

デバッガーの準備ができたら、次のセクションの説明に従ってサンプル・アプリケーションを実行します。

ステップ 3: デモ・アプリケーションの実行

ステップ 3.A: 行ブレークポイントの設定	ブレークポイントを設定する最も簡単な方法は、**[Source (ソース)]** ビューで青のドットの 1 つをダブルクリックすることです。青のドットは実行可能なコードを含む行を示し、ブレークポイントを設定できます。 1. **[Source (ソース)]** ビューで行 179 の青のドットをダブルクリックします。 178 179 while (1) { 180 //have something on the 青のドットがブレークポイント・アイコンに変わり、**[Console (コンソール)]** ビューにブレークポイントを説明するメッセージが表示されます。

ステップ 3.B: アプリケーションの実行	ツールバーの [Run (実行)] ボタンをクリックします。 プログラムがブレークポイントで停止し、プログラムカウンター ([Source (ソース)] ビューの黄色の矢印アイコンで示されます) が行 179 に設定されます。実行した行が青の背景色でハイライトされます。シミュレートされたターゲットのグラフィックス・コンソールに、アプリケーションの行 167 でプログラムされた文字列が表示されます。 次のステップに進むには、行 179 のブレークポイント・アイコンをダブルクリックして、この行のブレークポイントを削除します。
ステップ 3.C: 特定の関数の変数の評価	1. `screen_write_hex(unsigned int)` 関数にブレークポイントを設定します。 a. **[Debug (デバッグ)]** > **[Create Breakpoint... (ブレークポイントの作成...)]** をクリックします。 **[Create Breakpoint (ブレークポイントの作成)]** ダイアログボックスが開きます。 b. **[Browse (参照)]** をクリックします。 **[Symbol Browser (シンボルブラウザー)]** が開きます。ここでブレークポイントの場所を決定する式を選択できます。  c. `screen_write_hex(unsigned int)` 関数を選択して **[OK]** をクリックします。

	[Symbol Browser (シンボルブラウザー)] が閉じ、**[Create Breakpoint (ブレークポイントの作成)]** ダイアログボックスの **[Location: (場所:)]** フィールドにシンボルが表示されます。 d. **[Create Breakpoint (ブレークポイントの作成)]** ダイアログボックスで **[OK]** をクリックします。 ブレークポイントが設定されます。 2. [Run (実行)] ボタンをクリックします。 プログラムが実行されブレークポイントで停止します。 3. **[View (ビュー)]** > **[Locals (ローカル)]** を選択します。 **[Locals (ローカル)]** ビューが開きます。ここで関数のローカル変数を確認できます。 4. **[Step (ステップ)]** ボタンを数回クリックします。 デバッガーが関数を行単位でステップ実行します。プログラムを実行している間、ローカル変数の値がどのように変わるか確認します。
ステップ 3.D: デバッガーの調査	インテル® System Debugger のその他の機能を調べるには、以下の説明を参照してください。 1. **[Source Files (ソースファイル)]** ボタンをクリックして **[Source Files (ソースファイル)]** ビューを開きます。アプリケーションのビルドに使用されたすべてのソースが表示されます。 2. **[PCI]** ボタンをクリックして **[PCI Tool (PCI ツール)]** ダイアログボックスを開き、**[Scan (スキャン)]** ボタンをクリックします。ダイアログに実際のデータが表示されます。 3. **[GDT]** ボタンをクリックして **[GDT]** ビューを開きます。プロセッサーで使用されている現在のグローバル・ディスクリプター・テーブルが表示されます。エントリーを右クリックして **[Modify... (変更...)]** をクリックします。 **[Modify Descriptor (ディスクリプターの変更)]** ダイアログボックスが開き、関連する GDT ディスクリプターのセットアップが詳細に表示されます。 4. **[Assembler (アセンブラー)]** ボタンをクリックして **[Assembler (アセンブラー)]** ビューを開きます。現在の逆アセンブルされたコードが表示されます。 5. **[Callstack (コールスタック)]** ボタンをクリックして **[Callstack (コールスタック)]** ボタンを開きます。関数呼び出しのリストが表示されます。 6. **[Memory (メモリー)]** ボタンをクリックして **[Memory (メモリー)]** ビューを開きます。

	7. **[Paging (ページング)]** ボタンをクリックして **[Paging (ページング)]** ビューを開きます。右クリックして、ポップアップ・メニューから **[Translate Address (アドレスの変換)]** を選択します。 **[Address Translation (アドレス変換)]** ダイアログボックスが開きます。 **[Address/Symbolname (アドレス/シンボル名)]** フィールドに `start_kernel` と入力して **[OK]** をクリックすると、指定したシンボルのアドレス計算ステップが表示されます。 8. **[View (ビュー)]** > **[System Registers (システムレジスター)]** > **[Processor Specific Registers (プロセッサー固有のレジスター)]** を選択して [Processor Specific Registers (プロセッサー固有のレジスター)] ウィンドウを開きます。矢印をクリックして `64-Bit` エントリーを展開します。`RFL` エントリーをダブルクリックして **[Modify Register (レジスターの変更)]** ダイアログを開きます。 このレジスターのビットの値が説明とともに表示されます。関連するビットフィールドをクリックして値を編集できます。値を変更すると予測できない結果を引き起こすことがあるため、この機能は注意して使用してください。

関連資料

資料	説明
コードサンプルとチュートリアル (英語)	新しいユーザー向けに基本的な製品の機能を説明する、ダウンロード可能なサンプルコードとチュートリアル。サンプルコードは一般的なコードの非効率性を例示し、対応するチュートリアルはパフォーマンス解析製品を使用して問題を解決する方法を示します。
インテル® System Debugger — システムデバッグのドキュメント	インテル® System Debugger — システムデバッグに関する次のドキュメントが参考になります。 • インテル® System Debugger 2018 — システムデバッグ入門ガイド (英語) • インテル® System Debugger 2018 — システムデバッグ・ユーザー・リファレンス・ガイド (英語)
インテル® System Debugger — システムトレースのドキュメント	インテル® System Debugger — システムトレースに関する次のドキュメントが参考になります。 • インテル® System Debugger 2018 — システムトレース入門ガイド (英語) • インテル® System Debugger 2018 — システムトレース・ユーザーガイド (英語)

資料	説明
その他のドキュメント	最新の情報は、オンライン・ドキュメント (英語) を参照してください。 インテル® System Studio のすべてのドキュメントを含む ZIP ファイルを https://software.intel.com/en-us/articles/download-documentation-intel-system-studio-current-previous (英語) からダウンロードしてオフラインで利用できます。
インテル® System Studio 製品ページ	サポートおよび最新のドキュメントに関する情報

著作権と商標について

Intel、インテル、Intel ロゴ、VTune は、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

© Intel Corporation.

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。